ALTERNATIVE REPRESENTATIONS AND MULTIRECOMBINED APPROACHES FOR SOLVING THE SINGLE-MACHINE COMMON DUE DATE PROBLEM

Villagra A., Pandolfi D., De San Pedro M,. Vilanova G.
Proyecto UNPA-29/B017[1]
División Tecnología
Unidad Académica Caleta Olivia
Universidad Nacional de La Patagonia Austral
Ruta 3 Acceso Norte s/n
(9011) Caleta Olivia – Santa Cruz - Argentina
e-mail: {dpandolfi, edesanpedro,avillagra}@uaco.unpa.edu.ar; gvilanov@satlink.com
Phone/Fax : +54 0297 4854888

Gallard R.
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)[2]
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - Local 106
(5700) - San Luis -Argentina
e-mail: rgallard@unsl.edu.ar
Phone: +54 2652 420823
Fax : +54 2652 430224

ABSTRACT

Balance between exploitation and exploration is a main factor influencing convergence in an evolutionary algorithm. In order to improve this balance new trends in evolutionary algorithms make use of multi-recombinative approaches, known as *multiple-crossovers-on-multiple-parents* (MCMP). The use of a breeding individual (stud) which repeatedly mates individuals that randomly immigrates to a mating pool can further help the balance between exploration and exploitation.

For the single-machine common due date problem an optimal schedule is V-shaped around the due date. To produce V-shaped schedules an appropriate binary representation, associated with a schedule builder, can be used. In this representation each bit indicates if a corresponding job belongs either to the tardy or the non-tardy set. When contrasted with commonly used permutation representations this approach reduces the searching space from $n!$ to 2^n.

This paper compares three different implementations and shows their performance on a set of instances for the single machine scheduling problem with a common due date. Two of these approaches are based on a binary representation to form V-shaped schedules while the other is based on permutations. All these approaches apply different multirecombined methods. Details on implementation and results are discussed.

[1] The Research Group is supported by the Universidad Nacional de La Patagonia Austral.
[2] The LIDIC is supported by the Universidad Nacional de San Luis and the ANPCYT (National Agency to Promote Science and Technology).

1. INTRODUCTION

Among other heuristics [13], evolutionary algorithms (EAs) have been successfully applied to solve scheduling problems [14,15]. In EAs extreme exploitation can lead to premature convergence and intense exploration can make the search ineffective. In earlier works we devised a novel approach, *multiple-crossovers-per-couple* (MCPC), to allow multiple offspring per couple, as often happens in nature [8]. This approach allows us to deeply explore the recombination possibilities of previously found solutions. Implementation and results are discussed elsewhere [9], [10].To improve MCPC performance, by using the *multiparent approach* of Eiben [4], [5], [6], [7], the method was extended to MCMP where the multiple crossovers are applied to a set of multiple parents [12]. Results obtained in diverse single and multiobjective optimization problems indicated that the searching space is efficiently exploited by the multiple application of crossovers and efficiently explored by the greater number of samples provided by the multiple parents. A main property of this approach is revealed when observing the final population: all individuals are much more centred surrounding the optimum. This is an important issue when an application requires provision of multiple alternative near-optimal solutions.

According to Baker and Scudder [1] an optimal schedule for the earliness-tardiness problem in a single machine environment is V-shaped around the due date. Inserting problem-specific-knowledge, Lee and Kim [14] proposed a binary representation for a genetic algorithm which guarantees that all chromosome represents V-shaped schedules.

Following current trends and inserting problem-specific-knowledge this paper contrasts the behaviour of a multirecombined approach based in the use of a breeding individual (stud) [16], [17], [18] against a multirecombined approach based on uniform scanning crossover (USX). Next sections describe the earliness-tardiness scheduling problem, the new heuristics proposed and discuss the results obtained.

2. THE SINGLE-MACHINE COMMON DUE DATE PROBLEM

In scheduling, until recently, the mean tardiness criterion has been a standard method of measuring conformance to due dates ignoring the effects of jobs completing early. Just-in-time production emphasizes penalties to both early and tardy jobs giving rise to a nonregular performance measure and new methodologies in the design of solution procedures. In the restricted single-machine common due date problem, n jobs with deterministic processing times must be processed attempting to conform a common due date, thus minimizing penalties imposed to early and tardy jobs.

The problem can be stated as follows: A set of n jobs with deterministic processing times p_i and a common due date d is given. The jobs have to be processed on one machine. For each of the jobs an individual earliness α_i and tardiness b_i penalties are given. The goal is to find a schedule for the n jobs which jointly minimizes the sum of earliness and tardiness penalties.

Even simple in the formulation, this model leads to an optimization problem that is NP-Hard [3], and can be precisely stated as defined in [13]:

$$\min \sum_{i=1}^{n} (a_i E_i + b_i T_i) \text{ where}$$

$$E_i = \max\{0, d - c_i\},\ T_i = \max\{0, c_i - d\}$$

and c_i is the completion time of job j_i

3. ALTERNATIVE MULTIRECOMBINED APPROACHES

In our attempt to achieve a better balance between exploration and exploitation we devised MCMP-SRI, where a permutation based representation was adopted. Here, the process for creating offspring is performed as follows. From the old population an individual, designated the stud, is selected by means of proportional selection. The number of n_2 parents in the mating pool is completed with randomly created individuals (random immigrants). The stud mates every other parent, the couples undergo partial mapped crossover (PMX) and $2*n_2$ offspring are created. After crossover the best of these $2*n_2$ offspring is stored in a temporary children pool. The crossover operation is repeated n_1 times, until the children pool is completed. Finally, the best offspring created from n_2 parents and n_1 crossover is inserted in the new population. Children are not exposed to mutation because the random immigrants provide the necessary genetic diversity to the mating pool.

Attempting to insert problem-specific-knowledge we decided to follow Lee and Kim [14] proposal giving rise to MCMP-V. Regarding representation, a chromosome can suitably be represented by a binary string of n bits. Here, bit indicates one of two sets a corresponding job belongs to (1 represents the tardy job set T, and 0 represents the non-tardy job set E). For instance, if the chromosome is [0 0 1 0 1 1 1 1 0 0] the tardy set is formed as $T = \{ j_3, j_5, j_6, j_7, j_8 \}$ and the non-tardy set is formed as $E = \{ j_1, j_2, j_4, j_9, j_{10}\}$. Afterwards the jobs are sequenced in set E in non-increasing order of p_i / a_i and jobs in set T in non-decreasing order of p_i / b_i to form a V-shaped schedule. It is important to note that using the conventional permutation representation the algorithm is compelled to search in a problem space of size $n!$. Inserting problem-specific-knowledge through this representation the problem space size is reduced to 2^n. In MCMP-V, the process for creating the new population from the old population is performed as follows. Each time an offspring is to be inserted in the new population, n_2 parents selected from the old population undergo n_1 crossover operations (Uniform Scanning Crossover). Each crossover operation generates a single offspring, which eventually undergoes mutation. The Uniform Scanning Crossover (USX) can be safely used under this representation and is performed as follows: each gene in the child is provided from any of the corresponding alleles in the parents with equal probability. This method generates a single offspring. After crossover and mutation, offspring are stored in a temporary children pool. Finally, the best offspring created from n_2 parents and n_1 crossover is inserted in the new population. This process is repeated until the new population is completed.

Finally, MCMP-SRI-V is combination of both previously explained approaches, using binary representation and Uniform Crossover(UX). From the recombination point of view this method is similar to MCMP-SRI and from the representation point of view this method is similar to MCMP-V.

4. EXPERIMENTAL TESTS AND RESULTS

Instances from OR-library benchmarks [2, 3] for the common due date scheduling problem were selected to test all approaches. These benchmarks calculate the global due dates as

$$d = round\ [\text{SUM\_P} * h],$$

where *round[X]* gives the biggest integer which is smaller than or equal to X; *Sum_P* denotes the sum of the processing times of the n jobs and the parameter h is used to calculate more or less restrictive common due dates. Values for h are h=0.2; h=0.4; h=0.6 and h=0.8.

We performed a series of 10 runs for each of the 10 instances of the 10, 20 and 50 job problem sizes. For the 10 problem size, runs were performed for all h values. For the remaining problem sizes only h values of 0.2

and 0.4 were considered because inconsistencies were detected in the benchmark data. Some of them are indicated in table 1. Observe that for k = 1, 2, 3, 4, 5, 7, 8 and 10 the values for h = 0.6 and 0.8 are identical.

For MCMP-SRI, the parameters were set to convenient values as follows. Population sizes were fixed at 100 individuals for 10 and 20 job problems and at 150 for the 50 job problem. Probabilities for crossover were set to 0.65 in all experiments. The number of crossover n_1 and the number of parents n_2 were set to 6 and 8, respectively, for the 10 jobs problem and to 14 and 16, respectively, for the larger problems (20 and 50 jobs).

For MCMP-V and MCMP-SRI-V, the parameters were set to convenient values as follows. Population sizes and the maximum number of generations were fixed at 100 for all problem sizes. Probabilities for crossover were set to 0.65 in all experiments. The number n_1 of crossovers and the number n_2 of parents were set, respectively, to 6 and 8 for the 10 jobs problem size, to 14 and 16 for the 20 and 50 jobs problem sizes. Particularly, for MCMP-V approach the probability of mutation was fixed at 0.05.

Upper bounds for the 50 job examples:

n=50	h = 0.2	h = 0.4	h = 0.6	h = 0.8
k = 1	42,363	24,868	*17,990*	*17,990*
k = 2	33,637	19,279	*14,231*	*14,132*
k = 3	37,641	21,353	*16,497*	*16,497*
k = 4	30,166	17,495	*14,105*	*14,105*
k = 5	32,604	18,441	*14,650*	*14,650*
k = 6	36,920	21,497	14,251	14,075
k = 7	44,277	23,883	*17,715*	*17,715*
k = 8	46,065	25,402	*21,367*	*21,367*
k = 9	36,397	21,929	14,298	13,952
k = 10	35,797	20,048	*14,377*	*14,377*

Table 1. Example of upper bounds listed in the benchmarks

To compare the algorithms, the following relevant performance variables were chosen:

Ebest = ((best value - *opt_val*)/*opt_val*)100
It is the percentile error of the best-found individual when compared with the known, or estimated, optimum value *opt_val*. It gives us a measure on how far the best individual is from that *opt_val*.

Hit Ratio. It is the number of runs the algorithm found the optimum (or hits the upper bound) in a series of ten runs.

Gbest. It is the generation where the algorithm found the best individual.

Tables 2 to 4 summarize average values of the performance variables for all considered values of the restriction factor h (as above described for each problem size), over all 10 run series and the corresponding mean average through all instances.

Table 2, for 10 jobs problem size, shows that regarding quality of results all approaches find the optimal value in every run. Consequently, mean averages for *Ebest* and *Hit Ratio* are 0% and 1, respectively. These optimal values are found in average after 1.17 generations under MCMP-V, after 1.11 generations under MCMP-SRI-V and after 9.18 generations under MCMP-SRI. For this

problem size all known optimal values were reached and new optimal values were determined by MCMP-SRI for those instances where they were unknown.

Instances	MCMP-V			MCMP-SRI-V			MCMP-SRI		
	Ebest Avrg.	**Hit Ratio Avrg.**	**Gbest avrg.**	**Ebest Avrg.**	**Hit Ratio Avrg.**	**Gbest Avrg.**	**Ebest avrg.**	**Hit Ratio Avrg.**	**Gbest Avrg.**
sch10-1	0.00	1.00	1.13	0.00	1.00	1.13	0.00	1.00	9.10
sch10-2	0.00	1.00	1.10	0.00	1.00	1.13	0.00	1.00	7.68
sch10-3	0.00	1.00	1.13	0.00	1.00	1.05	0.00	1.00	10.13
sch10-4	0.00	1.00	1.18	0.00	1.00	1.10	0.00	1.00	8.85
sch10-5	0.00	1.00	1.13	0.00	1.00	1.20	0.00	1.00	9.70
sch10-6	0.00	1.00	1.08	0.00	1.00	1.03	0.00	1.00	9.48
sch10-7	0.00	1.00	1.23	0.00	1.00	1.09	0.00	1.00	9.30
sch10-8	0.00	1.00	1.35	0.00	1.00	1.10	0.00	1.00	9.20
sch10-9	0.00	1.00	1.20	0.00	1.00	1.05	0.00	1.00	9.35
sch10-10	0.00	1.00	1.23	0.00	1.00	1.23	0.00	1.00	8.98
Mean. Av.	**0.00**	**1.00**	**1.17**	**0.00**	**1.00**	**1.11**	**0.00**	**1.00**	**9.18**

Table 2. Performance variables values for 10 job problem size

As only upper bounds values are reported for the remaining problem sizes, a hit is recorded each time the algorithm hits (reaches or improves) the given upper bound.

Table 3, for 20 jobs problem size, shows that all approaches find better upper bounds with a general average improvement of 2.61% for MCMP-V, 2.49% for MCMP-SRI-V and 2.69% for MCMP-SRI. Mean average for *Hit Ratio* is 1 for MCMP-V and MCMP-SRI-V, while it is 0.99 for MCMP-SRI. These new upper bounds are found in average after 3.35 generations under MCMP-V, after 2.82 generations under MCMP-SRI-V and after 55.51 generations under MCMP-SRI..

Instances	MCMP-V			MCMP-SRI –V			MCMP-SRI		
	Ebest Avrg.	**Hit Ratio Avrg.**	**Gbest Avrg.**	**Ebest Avrg.**	**Hit Ratio Avrg.**	**Gbest Avrg.**	**Ebest avrg.**	**Hit Ratio Avrg.**	**Gbest Avrg.**
sch20-1	-0.42	1.00	3.60	-0.42	1.00	3.30	-0.41	0.95	64.30
sch20-2	-1.31	1.00	3.15	-1.31	1.00	3.05	-1.31	1.00	51.80
sch20-3	-1.54	1.00	2.55	-1.54	1.00	2.40	-1.54	1.00	56.80
sch20-4	-1.57	1.00	2.65	-1.57	1.00	2.45	-1.50	1.00	56.05
sch20-5	-2.92	1.00	3.10	-2.92	1.00	2.50	-2.92	1.00	52.30
sch20-6	-2.03	1.00	3.35	-2.03	1.00	3.50	-2.03	1.00	55.40
sch20-7	-3.85	1.00	5.95	-1.42	1.00	2.73	-3.66	1.00	59.90
sch20-8	-2.25	1.00	2.88	-3.51	1.00	2.70	-3.51	1.00	51.85
sch20-9	-0.95	1.00	3.20	-0.95	1.00	2.65	-0.83	1.00	53.10
sch20-10	-9.29	1.00	3.10	-9.29	1.00	2.95	-9.22	1.00	53.55
Mean. Av.	**-2.61**	**1.00**	**3.35**	**-2.49**	**1.00**	**2.82**	**-2.69**	**0.99**	**55.51**

Table 3. Performance variables values for 20 job problem size

Table 4, for 50 jobs problem size, shows the following results. MCMP-V finds better upper bounds, in every run, for all instances with a general average improvement of 5.13%, and a mean average *Hit Ratio* of 1. MCMP-SRI-V finds better upper bounds in every run for 9 out of 10 instances with a general average improvement of 4.25% and a mean average *Hit Ratio* of 0.97.

MCMP-SRI finds better upper bounds in every run for 5 out of 10 instances with a general average improvement of 1.93% and a mean average *Hit Ratio* of 0.65. This performance is achieved after 56.29, 58.21 and 99.21 generations, in average, for MCMP-V, MCMP-SRI-V and MCMP-SRI, respectively.

Further analysis on MCMP-V and MCMP-SRI-V show their *effectiveness* in the 20 and 50 job problem sizes. Here the number of times when the upper bound was improved, attained or not attained is recorded. Recall that for these problem sizes the algorithm was run for each instance with *h* values of 0.2 and 0.4.

Instances	MCMP-V			MCMP-SRI -V			MCMP-SRI		
	Ebest Avrg.	Hit Ratio Avrg.	Gbest Avrg.	Ebest Avrg.	Hit Ratio Avrg.	Gbest avrg.	Ebest avrg.	Hit Ratio Avrg.	Gbest Avrg.
sch50-1	-4.10	1.00	52.70	-3.25	1.00	61.20	0,27	0,30	101,80
sch50-2	-8.02	1.00	50.10	-7.22	1.00	59.95	-4,45	1,00	105,60
sch50-3	-6.23	1.00	54.20	-5.13	1.00	57.00	-4,05	1,00	97,00
sch50-4	-6.32	1.00	62.40	-5.37	1.00	56.30	-8,64	1,00	105,70
sch50-5	-1.58	1.00	52.10	-0.09	0.65	58.25	3,63	0,00	96,70
sch50-6	-5.08	1.00	57.10	-5.23	1.00	60.58	-0,65	0,70	90,90
sch50-7	-3.05	1.00	52.75	-2.02	1.00	55.40	0,74	0,00	89,50
sch50-8	-3.40	1.00	54.70	-2.29	1.00	59.40	-0,08	0,50	82,20
sch50-9	-7.39	1.00	65.20	-6.68	1.00	57.85	-2,59	1,00	111,20
sch50-10	-6.12	1.00	61.65	-5.23	1.00	62.15	-3,45	1,00	111,50
Mean. Av.	**-5.13**	**1.00**	**56.29**	**-4.25**	**0.97**	**58.81**	**-1,93**	**0,65**	**99,21**

Table 4. Performance variables values for 50 job problem size

Problem size	Improve upper bound	Attain upper bound	Do not attain upper bound	Total of cases
20	19	1	0	20
50	20	0	0	20
Total	**39**	**1**	**0**	**40**
Percent	**97.5**	**2.5**	**0.0**	**100**

Table 6. Effectiveness of MCMP-V and MCMP-SRI-V over all experiments for larger problem sizes.

5. CONCLUSIONS

In this work we report three alternative multirecombined approaches with different representation. By means of stud MCMP-SRI attempts to find equilibrium between exploration and exploitation in the search process. MCMP-V reduces the size of problem space from n! to 2^n by inserting problem-specific-knowledge (an optimality condition). MCMP-SRI-V attempts to combine both objectives.

All MCMP variants considered here showed outstanding behaviour when facing this difficult earliness-tardiness scheduling problem by providing new optimal solution for smaller instances and improving the upper bound in the larger instances.

Those approaches which contemplated the optimality condition behaved better than MCMP-SRI reaching an effectiveness of 97.5% when improving reported upper bounds. Nevertheless MCMP-SRI behaves better than other previous tested EAs.

Future work will be devoted to investigate new multirecombined variants, insertion of problem-specific-knowledge, dynamic control and self-adaptation of parameters to follow this trend and test them in the larger benchmarks of the OR library.

6. REFERENCES

[1] Baker K. and Scudder G., Sequencing with earliness and tardiness penalties: A review, Operations Research, Vol. 38, pp 22-36, 1990.

[2] Beasley J. E. Common Due Date Scheduling, OR Library, http://mscmga.ms.ic.ac.uk/jeb/orlib.

[3] Biskup D. and Feldmann M., Benchmarks for scheduling on a single-machine against restrictive and unrestrictive common due dates' , Discussion Paper No. 397, August 1998, University of Bielefeld.

[4] Chen T. and Gupta M., Survey of scheduling research involving due date determination decision, European Journal of Operational Research, vol 38, pp. 156-166, 1989.

[5] Eiben A.E., Raué P-E., and Ruttkay Zs., *Genetic algorithms with multi-parent recombination*. In Davidor, H.-P. Schwefel, and R. Männer, editors, Proceedings of the 3rd Conference on Parallel Problem Solving from Nature, number 866 in LNCS, pages 78-87. Springer-Verlag, 1994

[6] Eiben A.E., van Kemenade C.H.M., and Kok J.N., *Orgy in the computer: Multi-parent reproduction in genetic algorithms*. In F. Moran, A. Moreno, J.J. Merelo, and P. Chacon, editors, Proceedings of the 3rd European Conference on Artificial Life, number 929 in LNAI, pages 934-945. Springer-Verlag, 1995.

[7] Eiben A.E. and. Bäck Th., An empirical investigation of multi-parent recombination operators in evolution strategies. Evolutionary Computation, 5(3):347-365, 1997.

[8] Esquivel S., Leiva A., Gallard R., - *Multiple Crossover per Couple in Genetic Algorithms,* Proceedings of the Fourth IEEE Conference on Evolutionary Computation (ICEC'97), pp 103-106, ISBN 0-7803-3949-5, Indianapolis, USA, April 1997.

[9] Esquivel S., Gallard R. and Michalewicz Z., *MCPC: Another Approach to Crossover in Genetic Algorithms,* Proceeding of Primer Congreso Argentino de Ciencias de la Computación, pp 141 - 150, 1995.

[10] Esquivel S., Leiva A., Gallard R., - *Multiple crossover per couple in genetic algorithms*. Proc. of the 4th IEEE International Conf. on Evolutionary Computation (ICEC'97), pp 103-106, Indianapolis, USA, April 1997.

[11] Esquivel S., Leiva A., Gallard R.: *Couple Fitness Based Selection with Multiple Crossover per Couple in Genetic Algorithms.* Proceedings of the International Symposium on Engineering of Intelligent Systems (EIS´98), La Laguna, Tenerife, Spain Vol. 1, pp 235-241, ed. E.Alpaydin. Publishesd by ICSC Academic Press, Canada/Switzerland, February 1998.

[12] Esquivel S., Leiva H.,.Gallard R., *Multiple crossovers between multiple parents to improve search in evolutionary algorithms*, Proceedings of the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, pp 1589-1594.

[13] Hall N. and Posner M., Earliness Tardiness Scheduling problems: weighted deviation of completion times about a common due date, Operations Research, vol. 39, pp. 836-846, 1991.

[14] Lee C. and Kim S., Parallel Genetic Algorithms for the tardiness job scheduling problem with general penaltiy weights, International Journal of Computers and Industrial Engineering, vol. 28, pp. 231-243, 1995

[15] Michalewicz, M., *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, third revised edition, 1996.

[16] Pandolfi D., Vilanova G., De San Pedro M., Villagra A. Multirecombining studs and immigrants in evolutionary algorithm to face earliness-tardiness scheduling problems. Proceedings of the International Conference in Soft Computing. University of Paisley, Scotland, U.K., June 2001.

[17] Pandolfi D., Vilanova G., De San Pedro M., Villagra A. Solving the single-machine common due date problem via studs and immigrants in Evolutionary Algorithms. Proceedings of the Multiconference on Systemics, Cybernetics and informatics. Orlando, Florida July 2001.

[18] Pandolfi D., Vilanova G., De San Pedro M., Villagra A. Adaptability of Multirecombined Evolutionary Algorithms in the single-machine common due date problem. Proceedings of the Multiconference on Systemics, Cybernetics and informatics. Orlando, Florida July 2001.